Model-agnostic Approaches to Handling Noisy Labels When Training Sound Event Classifiers

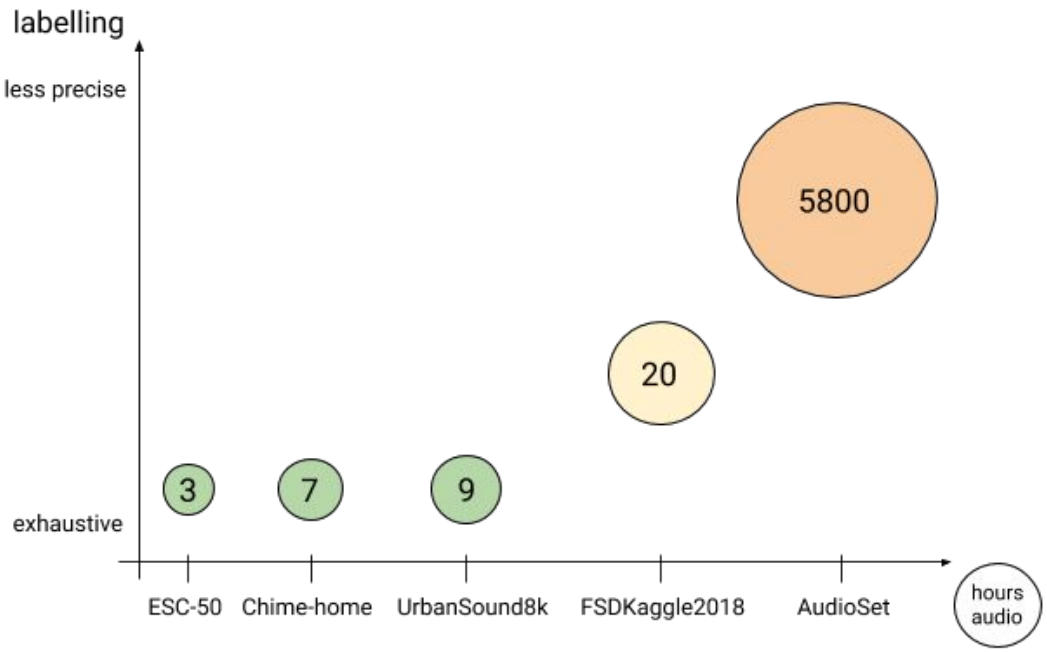
Eduardo Fonseca, Frederic Font, and Xavier Serra

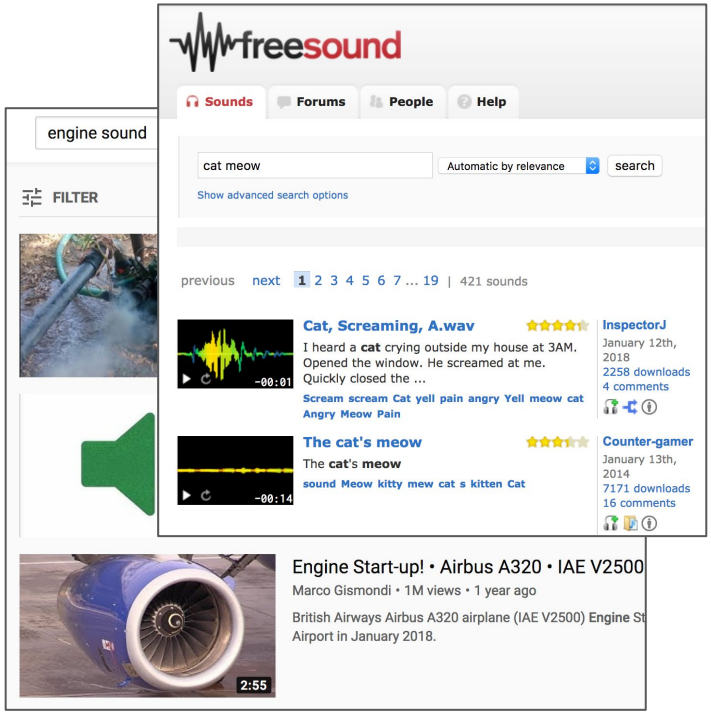
upf. Universitat Pompeu Fabra Barcelona

MTG Music Technology Group

Label noise in sound event classification

- Labels that fail to properly represent acoustic content in audio clip
- Why is label noise relevant?





- Label noise effects: performance decrease / increased complexity

Our use case

- **Given a learning pipeline:**
  - → sound event dataset with noisy labels & deep network
  - → that we do not want to change
    - no network modifications / no additional (clean) data

- **How can we improve performance in THIS setting?**
  - → just minimal changes

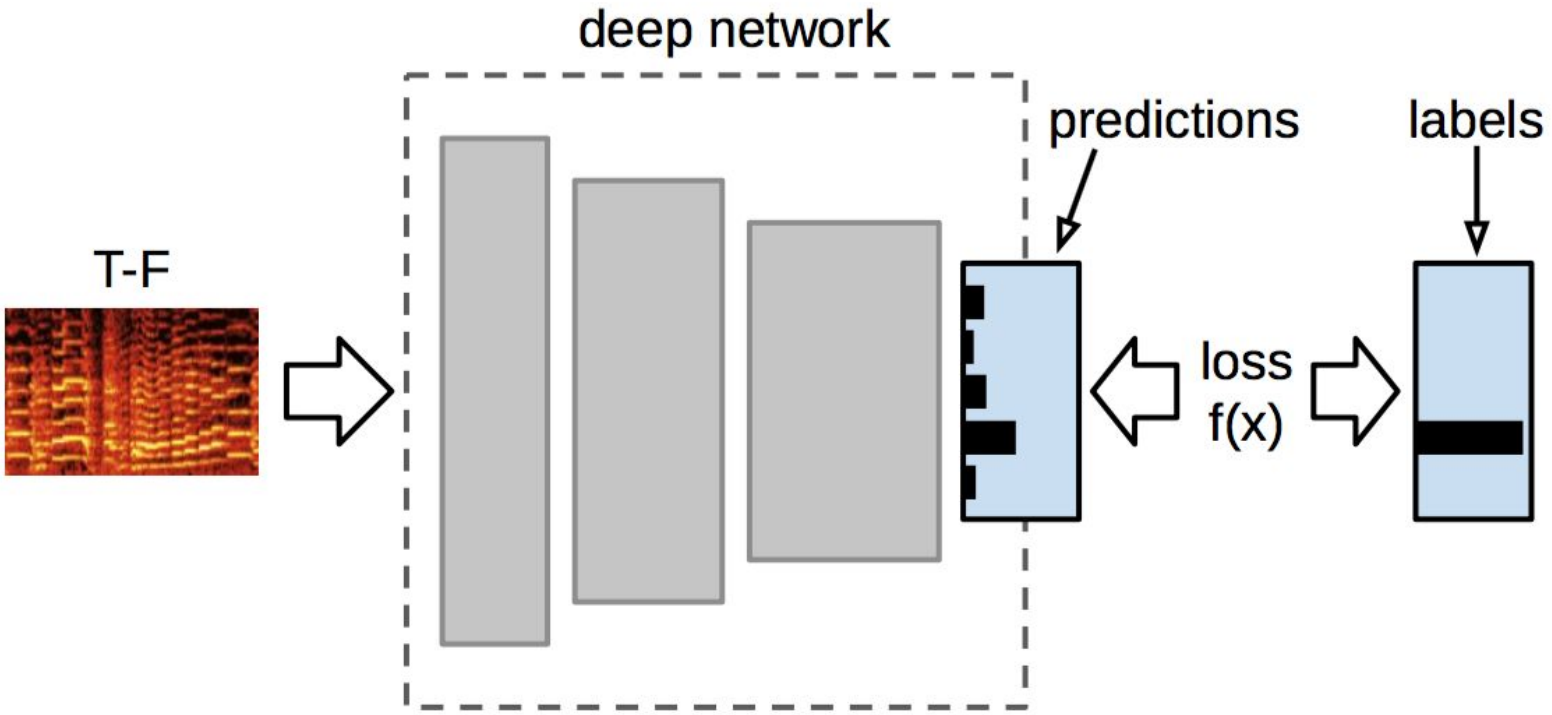
Our use case

- **Given a learning pipeline**
  - sound event dataset with noisy labels & deep network
  - that we do not want to change
    - no network modifications / no additional (clean) data

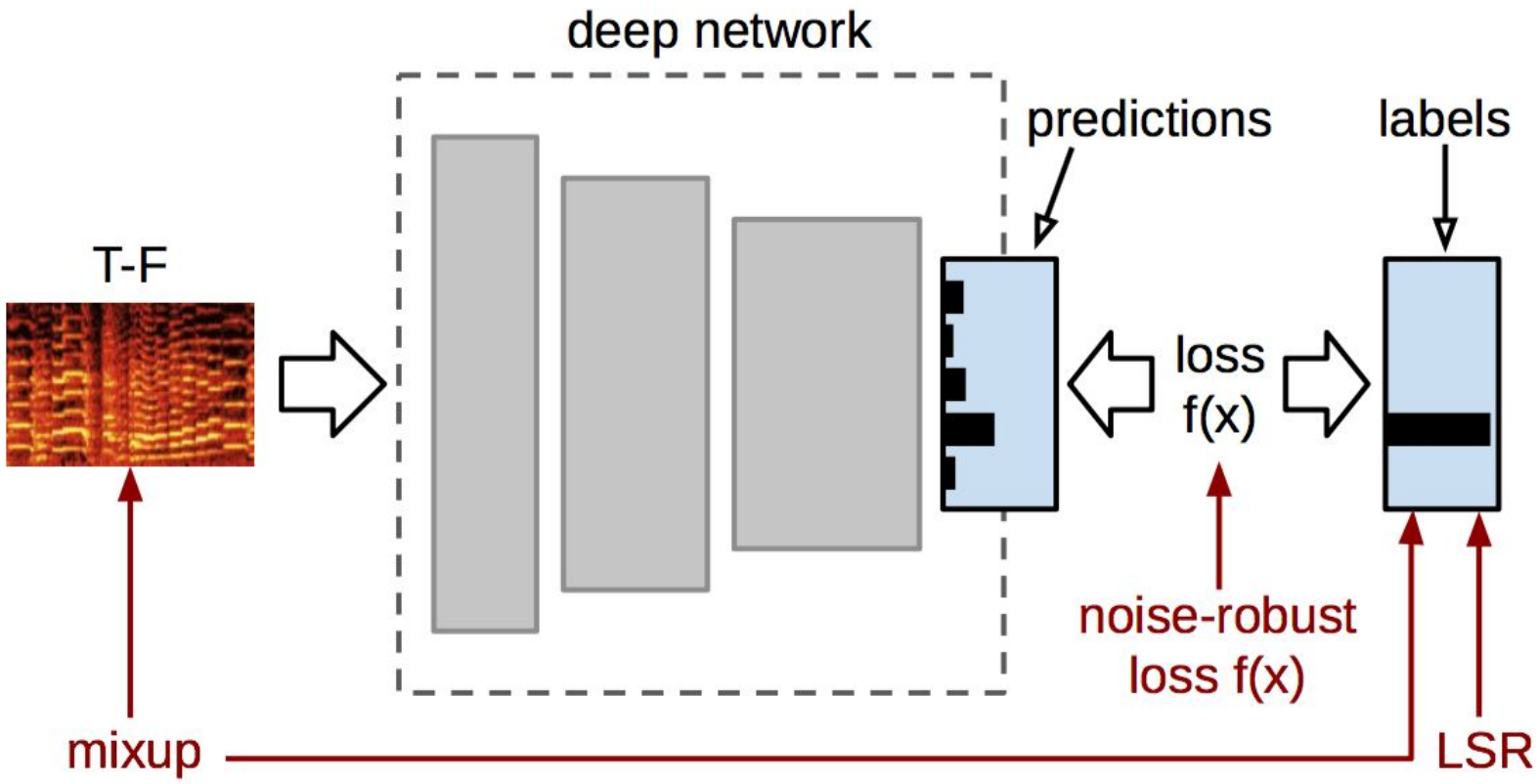
- **How can we improve performance in THIS setting?**
  - just minimal changes

- **Our work**
  - simple & efficient ways to boost performance in presence of noisy labels
  - agnostic to network architecture
  - that can be plugged into existing learning settings

Our use case

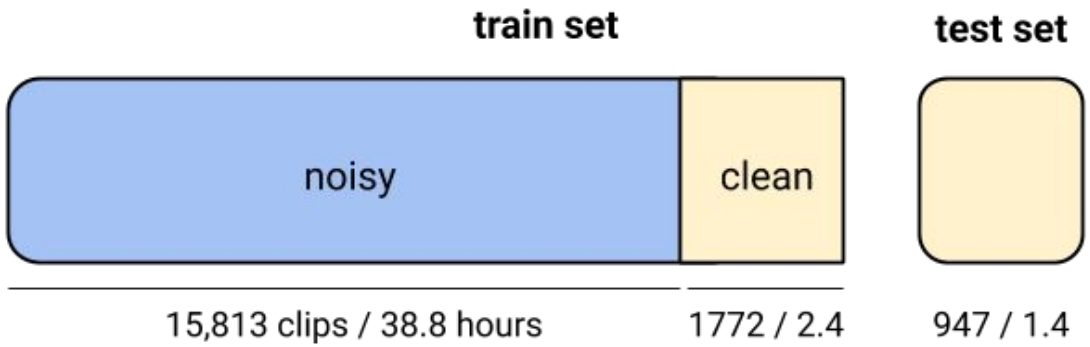


Our use case

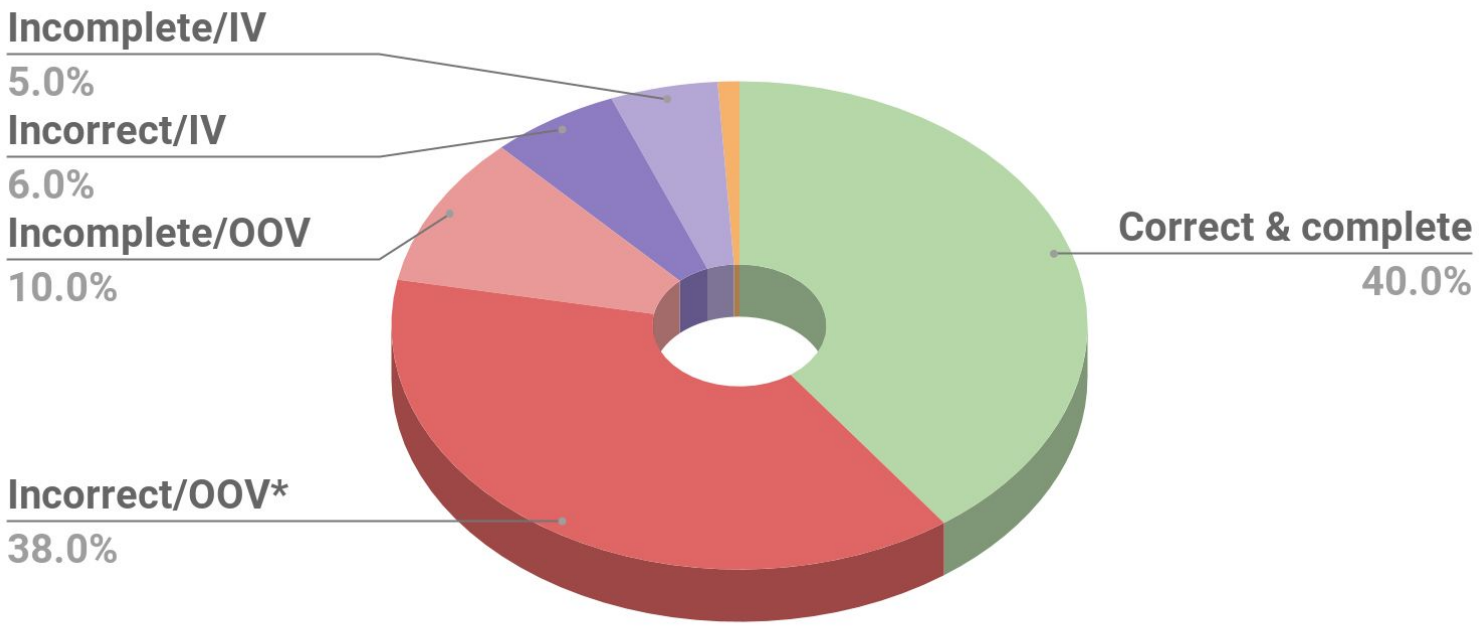

deep network
predictions
labels
T-F
loss
f(x)
noise-robust
loss f(x)
mixup
LSR

Dataset: FSDnoisy18k

- Freesound audio organized with 20 class labels from AudioSet Ontology
- audio content retrieved by user-provided tags
  - → per-class varying degree of types and amount of label noise
- 18k clips / 42.5 h
- singly-labeled data -> multi-class problem
- variable clip duration: 300ms - 30s
- proportion **train_noisy** / train_clean = 90% / 10%
- freely available http://www.eduardofonseca.net/FSDnoisy18k/

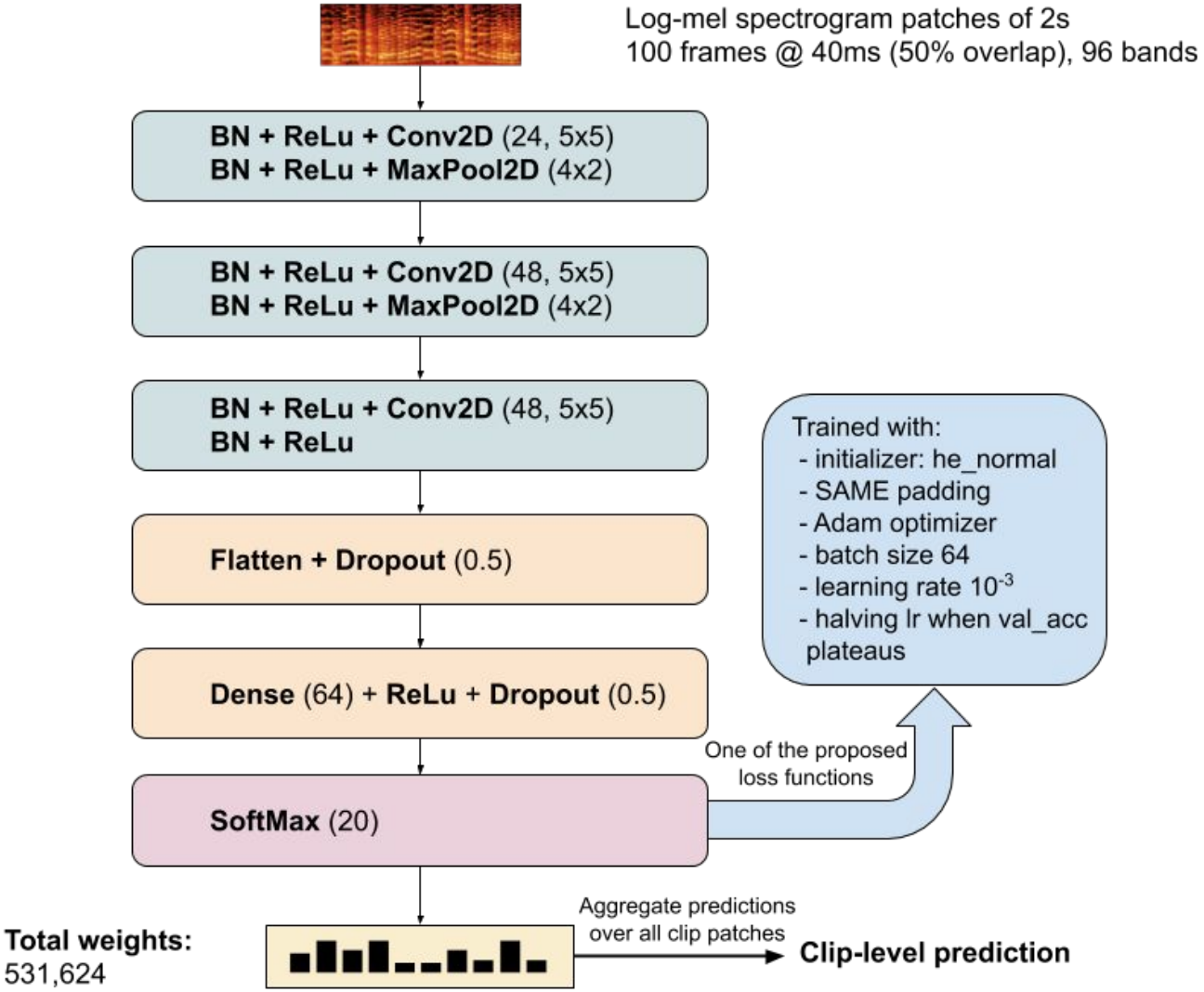


Label noise distribution in FSDnoisy18k



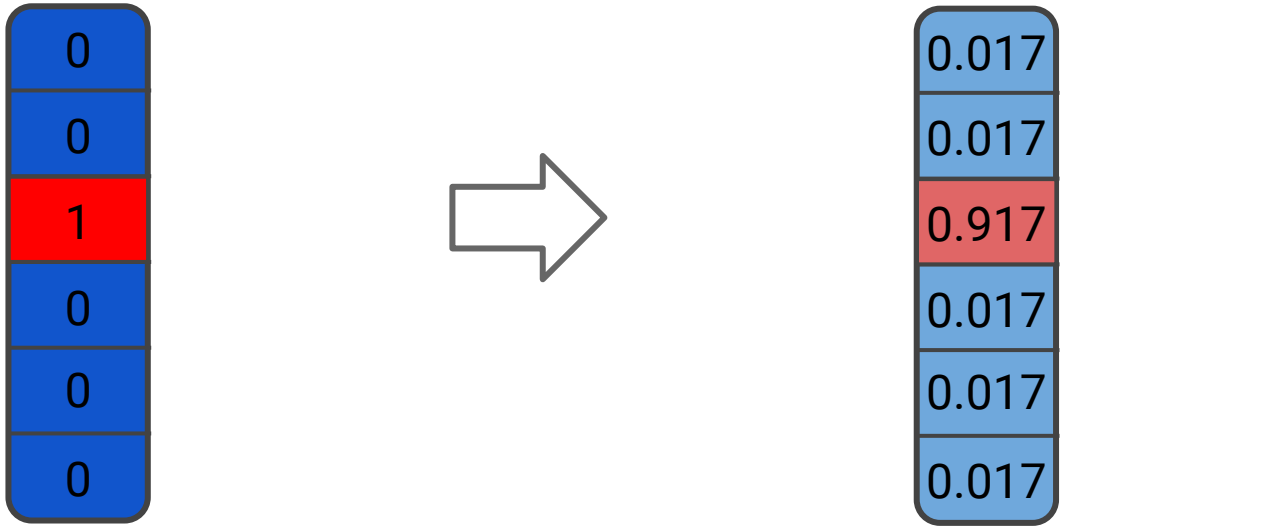
- IV: in-vocabulary, events that are part of our target class set
- OOV: out-of-vocabulary, events not covered by the class set

CNN baseline system



Label Smoothing Regularization (LSR)

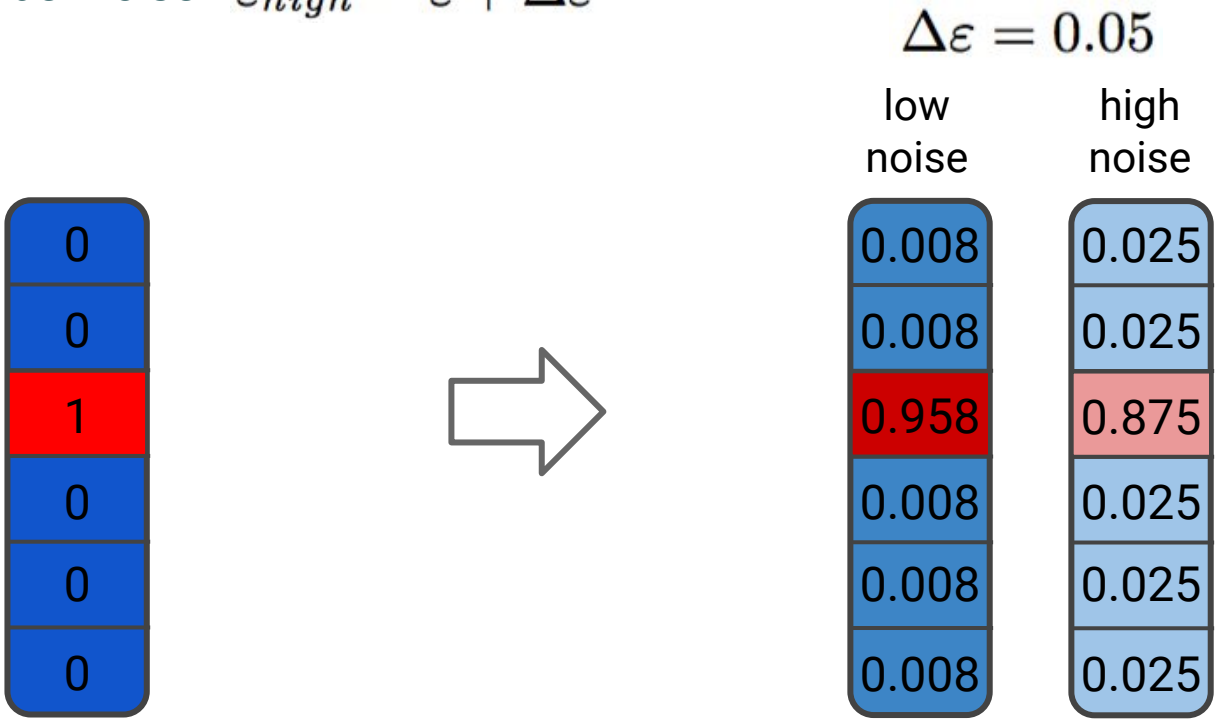
- Regularize the model by promoting less confident output distributions
  - → smooth label distribution: hard → soft targets



Noise dependent LSR

- Encode prior of label noise: 2 groups of classes:
  - low label noise $\varepsilon_{low} = \varepsilon - \Delta\varepsilon$
  - high label noise $\varepsilon_{high} = \varepsilon + \Delta\varepsilon$

$\Delta\varepsilon = 0.05$



LSR results

- Vanilla LSR provides limited performance
- Better by encoding prior knowledge of label noise through noise-dependent epsilon

Table 2: Average classification accuracy (%) and 95% confidence interval (7 runs) obtained by LSR incorporated to the baseline system.

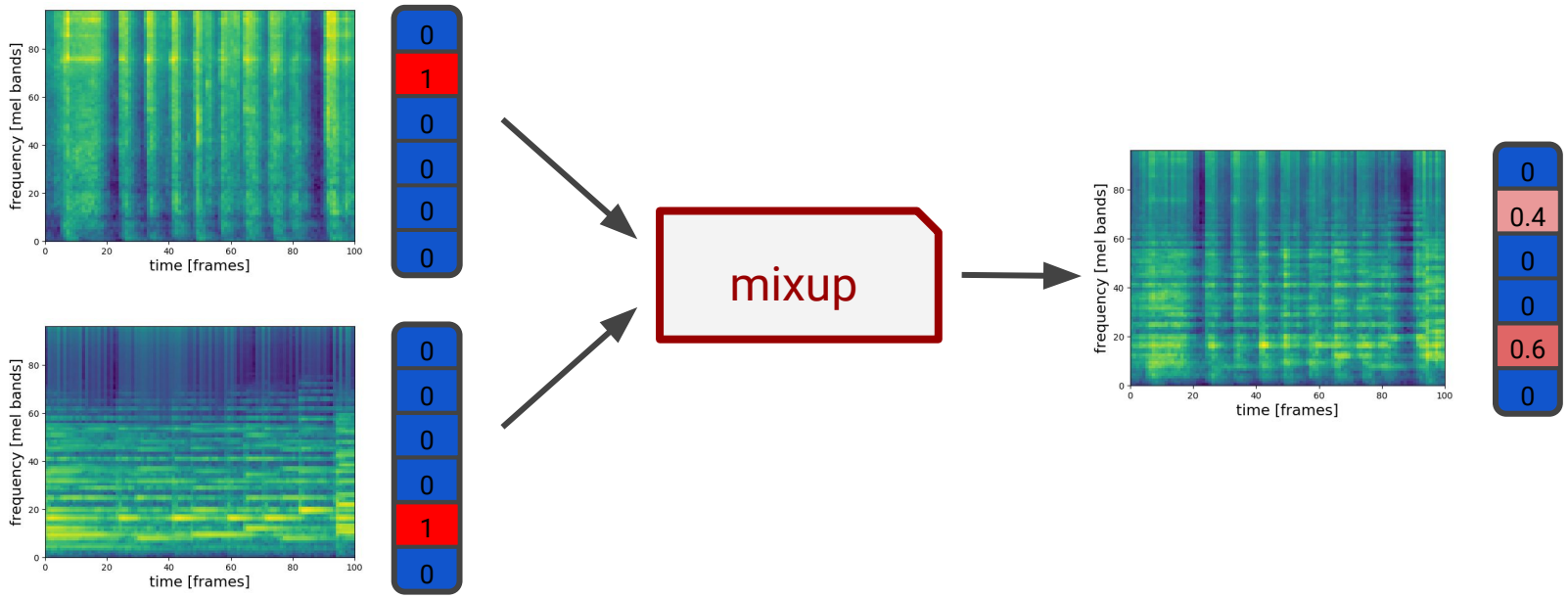
Approach	**Accuracy**
Baseline	66.5 ± 0.6
LSR ($\varepsilon = 0.1$)	66.8 ± 1.0
LSR ($\varepsilon = 0.15$)	67.1 ± 1.1
LSR ($\varepsilon = 0.15 \pm 0.05$)	68.1 ± 0.8

mix-up

- Linear interpolation
  - → in the feature space $\tilde{x} = \lambda x_i + (1 - \lambda) x_j$
  - → in the label space $\tilde{y} = \lambda y_i + (1 - \lambda) y_j$

$$\lambda \sim Beta(\alpha, \alpha)$$
$$\lambda \in [0, 1]$$

- Again, soft targets



mix-up results

- mix-up applied from the beginning: limited boost
- creating virtual examples far from the training distribution confuses the model
- warming-up the model helps!

Table 2: Average classification accuracy (%) and 95% confidence interval (7 runs) obtained by LSR and mixup incorporated to the baseline system.

Approach	**Accuracy**
Baseline	66.5 ± 0.6
LSR ($\varepsilon = 0.1$)	66.8 ± 1.0
LSR ($\varepsilon = 0.15$)	67.1 ± 1.1
LSR ($\varepsilon = 0.15 \pm 0.05$)	68.1 ± 0.8
mixup ($\alpha = 0.1$)	67.1 ± 0.8
mixup ($\alpha = 0.2$)	66.6 ± 0.7
warm-up (10 epochs) & mixup ($\alpha = 0.3$)	$\mathbf{68.4} \pm 0.5$

Noise-robust loss function

Noise-robust loss function

- Default loss function in multi-class setting: Categorical Cross-Entropy (CCE)

$$\mathcal{L}_{cce} = -\sum_{k=1}^{K} y(k) \log(p(k))$$

predictions

target labels

Noise-robust loss function

- Default loss function in multi-class setting: Categorical Cross-Entropy (CCE)

$$\mathcal{L}_{cce} = -\sum_{k=1}^{K} y(k) \log(p(k))$$

- CCE is sensitive to label noise: emphasis on *difficult* examples (weighting)
  - → beneficial for clean data
  - → detrimental for noisy data

Noise-robust loss function

- $\mathcal{L}_q$ loss intuition
  - CCE: sensitive to noisy labels (weighting)
  - Mean Absolute Error (MAE): $\mathcal{L}_{mae} = \sum_{k=1}^{K} |y(k) - p(k)|$
    - avoid weighting
    - difficult convergence

Zhilu Zhang and Mert Sabuncu, ***Generalized cross entropy loss for training deep neural networks with noisy labels***. In NeurIPS 2018

Noise-robust loss function

- $\mathcal{L}$q loss intuition
  - CCE: sensitive to noisy labels (weighting)
  - Mean Absolute Error (MAE): $\mathcal{L}_{mae} = \sum_{k=1}^{K} |y(k) - p(k)|$
    - avoid weighting
    - difficult convergence
- $\mathcal{L}$q loss is a generalization of CCE and MAE:
  - negative Box-Cox transformation of softmax predictions

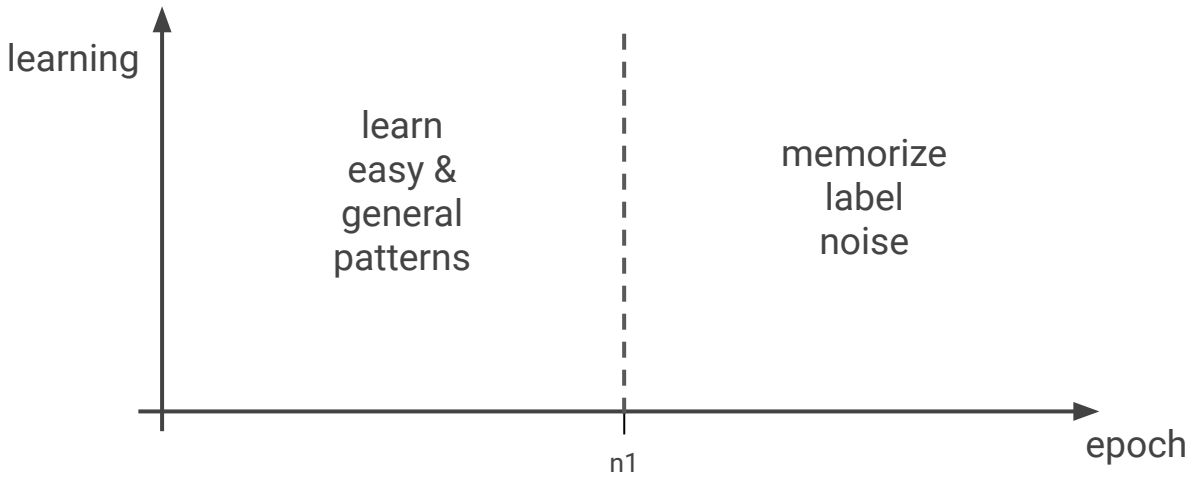
$$\mathcal{L}_q = \frac{1 - (\sum_{k=1}^{K} y(k)p(k))^q}{q}, \quad q \in (0, 1]$$

  - $q = 1 \rightarrow \mathcal{L}$q = MAE ; $q \rightarrow 0 \rightarrow \mathcal{L}$q = CCE

Zhilu Zhang and Mert Sabuncu, ***Generalized cross entropy loss for training deep neural networks with noisy labels***. In NeurIPS 2018

Learning and noise memorization

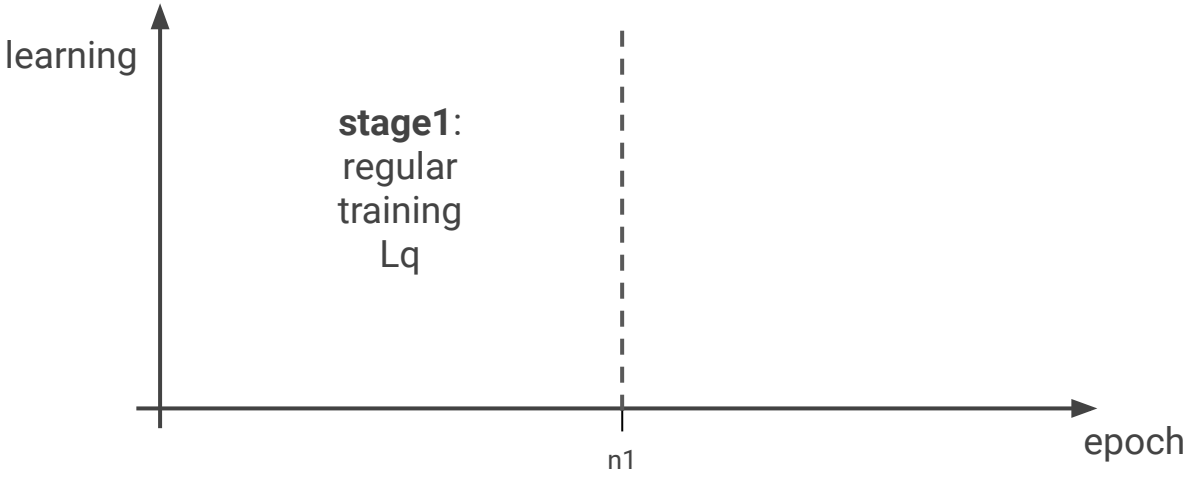
- Deep networks in presence of label noise
  - → problem is more severe as learning progresses



Arpit, Jastrzebski, Ballas, Krueger, Bengio, Kanwal, Maharaj, Fischer, Courville, and Bengio., ***A closer look at memorization in deep networks***. In ICML 2017

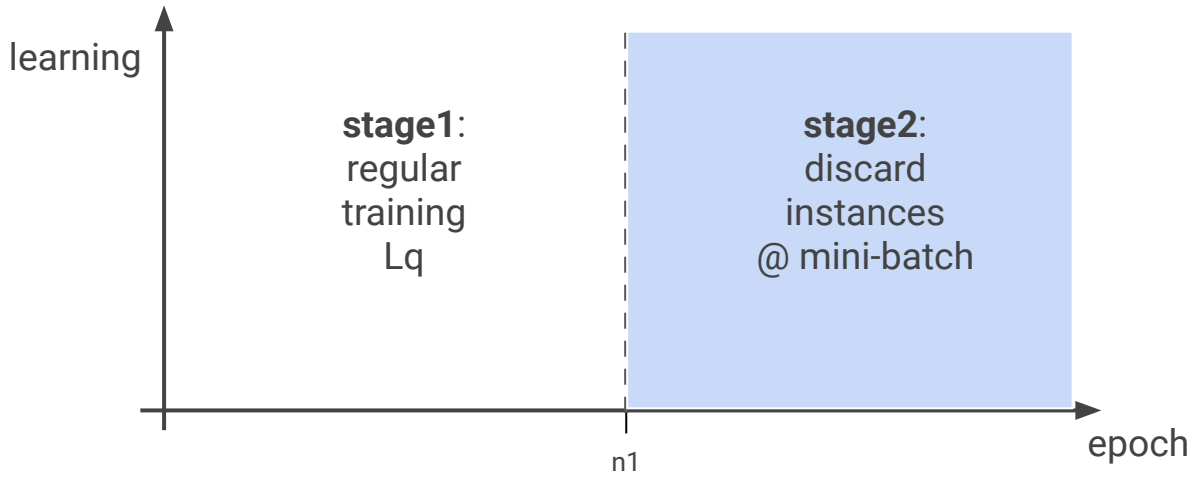
Learning as a two-stage process

- Learning process as a two-stage process
- After n1 epochs:
  - → model has converged to some extent
  - → use it for instance selection
    - identify instances with large training loss
    - ignore them for gradient update



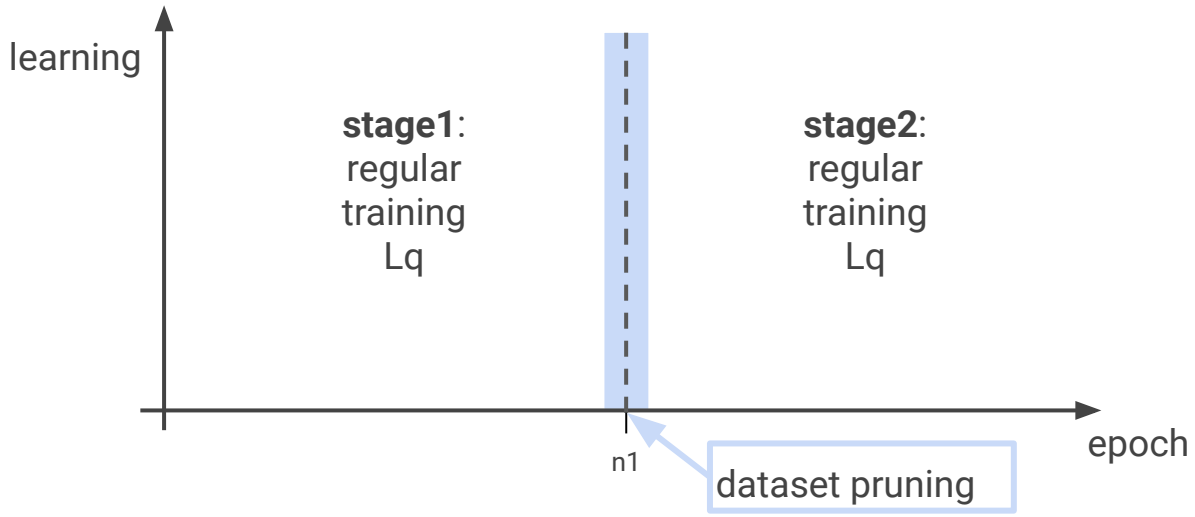
Ignoring large loss instances

- Approach 1:
  - **discard** large loss instances **from each mini-batch** of data
  - dynamically at every iteration
  - time-dependent loss function



Ignoring large loss instances

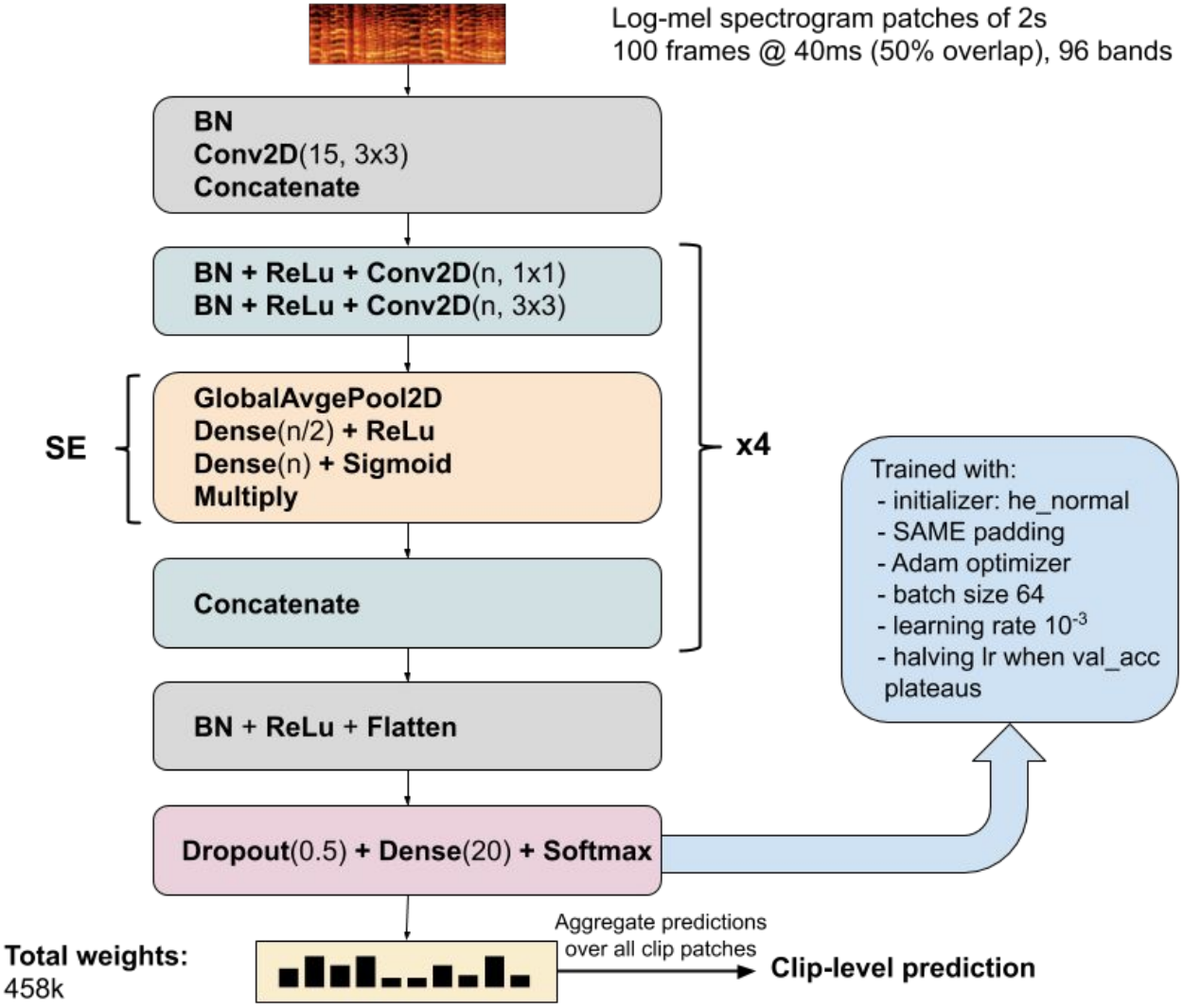
- Approach 2:
  - use checkpoint to predict scores on whole dataset
  - convert to loss values
  - **prune dataset**, keeping a subset to continue learning



Noise-robust loss function results

- We report results with two models
  - using baseline
  - using a more accurate model

A more accurate model: DenSE



Noise-robust loss function results

- pruning dataset slightly outperforms discarding at mini-batch

Table 4: Average classification accuracy (%) and 95% confidence interval (7 runs) obtained by loss function approaches.

Approach	**Baseline**	**DenSE**
CCE	66.5 ± 0.6	67.9 ± 0.7
$\mathcal{L}_q$	68.4 ± 0.5	69.2 ± 0.8
$\mathcal{L}_{q,discard}$ (discard at mini-batch)	68.8 ± 0.9	69.8 ± 0.7
$\mathcal{L}_{q,prune}$ (prune dataset)	$\mathbf{69.0} \pm 0.6$	$\mathbf{70.2} \pm 0.5$

Noise-robust loss function results

- pruning dataset slightly outperforms discarding at mini-batch
- discarding at mini-batch is less stable

Table 4: Average classification accuracy (%) and 95% confidence interval (7 runs) obtained by loss function approaches.

Approach	**Baseline**	**DenSE**
CCE	66.5 ± 0.6	67.9 ± 0.7
$\mathcal{L}_q$	68.4 ± 0.5	69.2 ± 0.8
$\mathcal{L}_{q,discard}$ (discard at mini-batch)	68.8 ± 0.9	69.8 ± 0.7
$\mathcal{L}_{q,prune}$ (prune dataset)	$\mathbf{69.0} \pm 0.6$	$\mathbf{70.2} \pm 0.5$

Noise-robust loss function results

- pruning dataset slightly outperforms discarding at mini-batch
- discarding at mini-batch is less stable
- DenSE:
  - → higher boosts wrt $\mathcal{L}_q$
  - → more stable

Table 4: Average classification accuracy (%) and 95% confidence interval (7 runs) obtained by loss function approaches.

Approach	**Baseline**	**DenSE**
CCE	66.5 ± 0.6	67.9 ± 0.7
$\mathcal{L}_q$	68.4 ± 0.5	69.2 ± 0.8
$\mathcal{L}_{q,discard}$ (discard at mini-batch)	68.8 ± 0.9	69.8 ± 0.7
$\mathcal{L}_{q,prune}$ (prune dataset)	$\mathbf{69.0} \pm 0.6$	$\mathbf{70.2} \pm 0.5$

Summary & takeaways

- Three simple model agnostic approaches against label noise
  - → easy to incorporate to existing pipelines
  - → minimal computational overhead
  - → absolute accuracy boosts ~ 1.5 - 2.5%

- Most promising: pruning dataset using model as instance selector
  - → could be done several times iteratively
  - → useful for dataset cleaning
  - → but dependent on pruning time & pruned amount

Model-agnostic Approaches to Handling Noisy Labels When Training Sound Event Classifiers

Thank you!

https://github.com/edufonseca/waspaa19

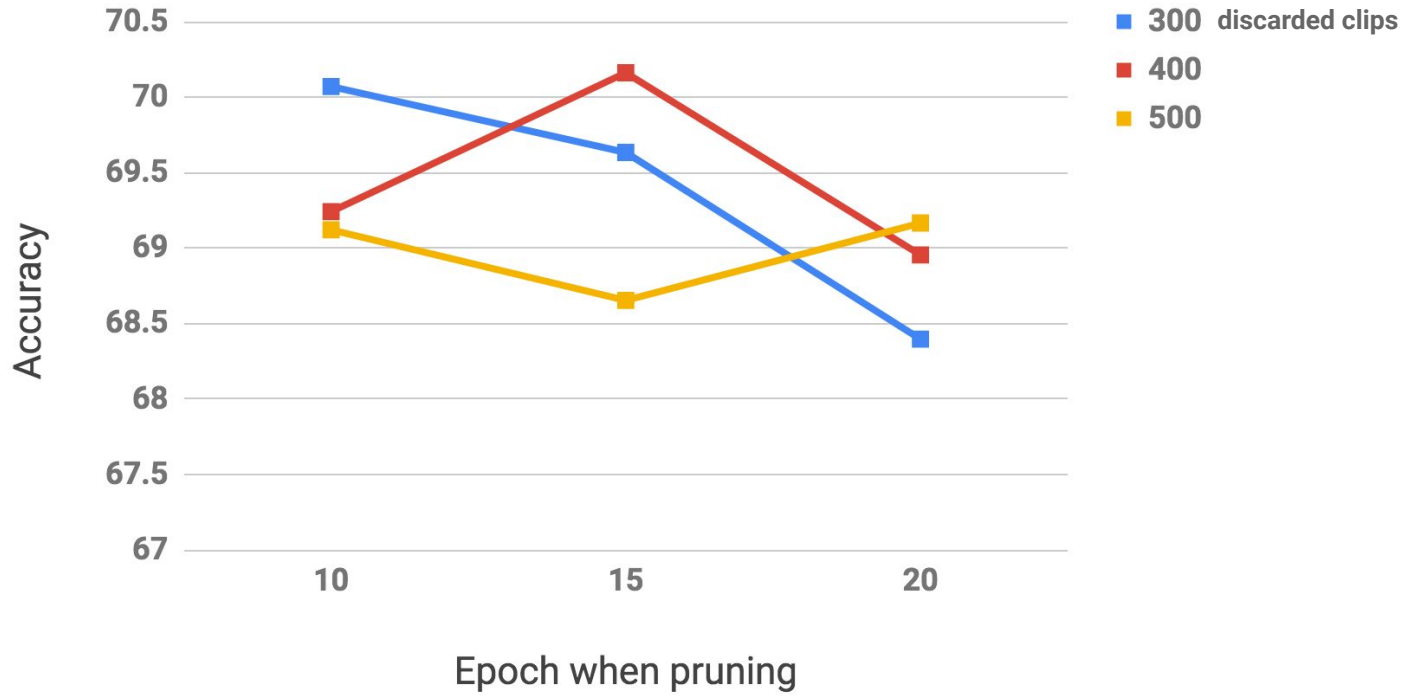
Eduardo Fonseca, Frederic Font, and Xavier Serra

upf. Universitat Pompeu Fabra Barcelona

MTG Music Technology Group

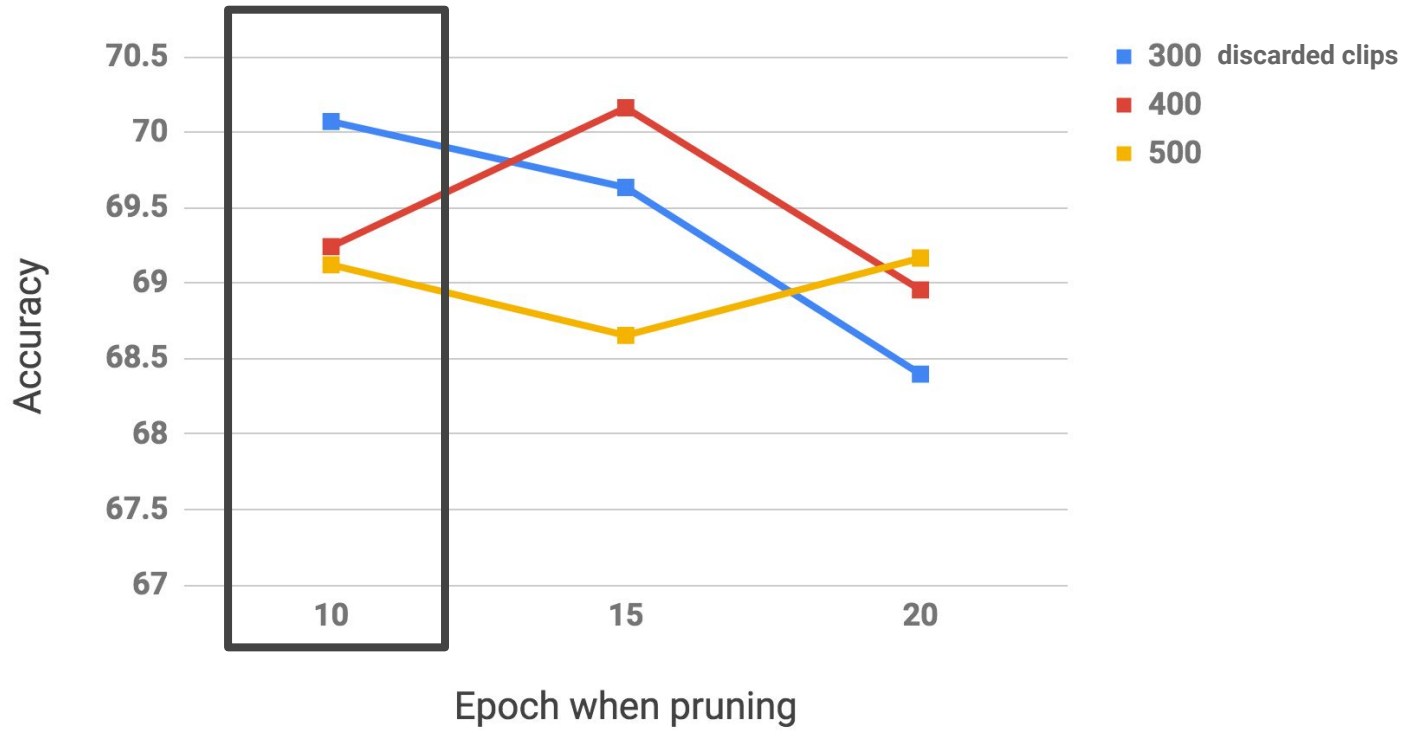
Dataset pruning & noise memorization

- We explore pruning the dataset at different epochs



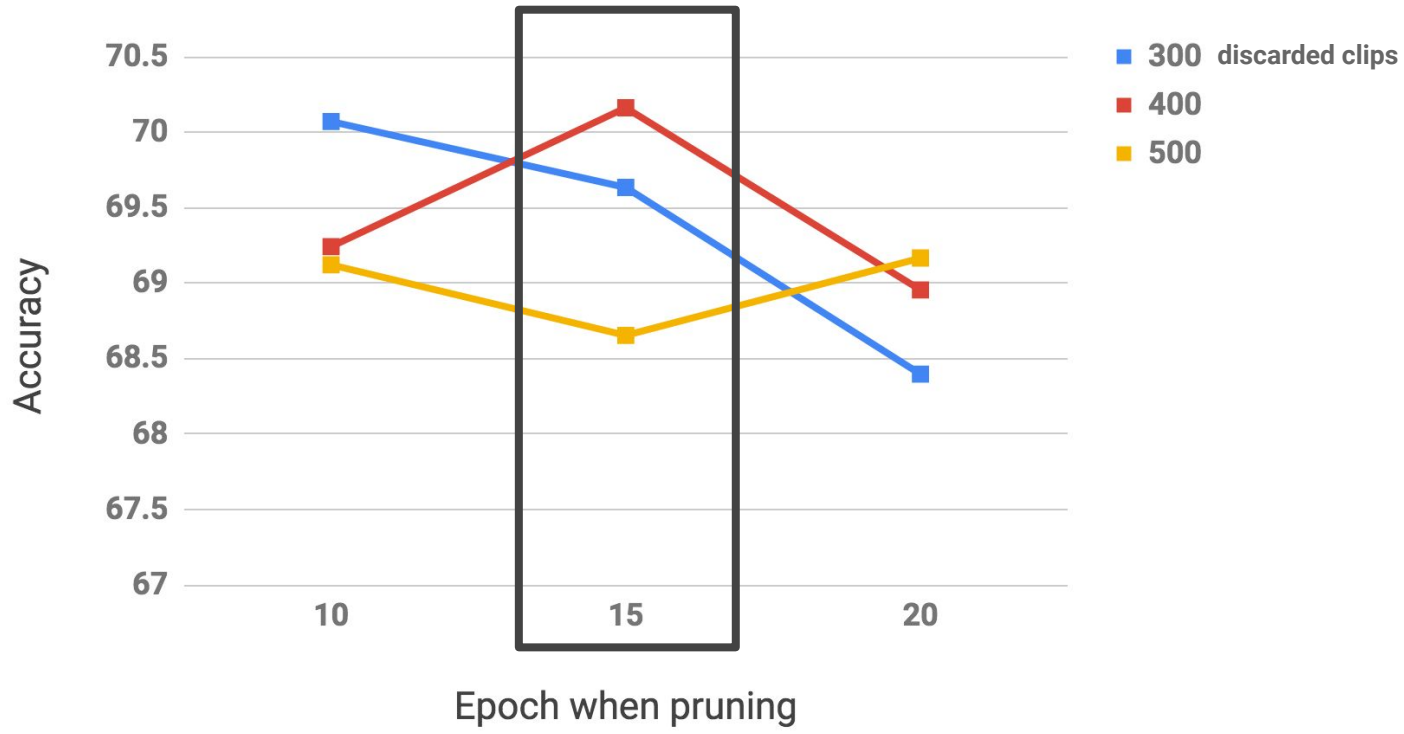
Dataset pruning & noise memorization

- model not too accurate ➔ pruning many clips is worse



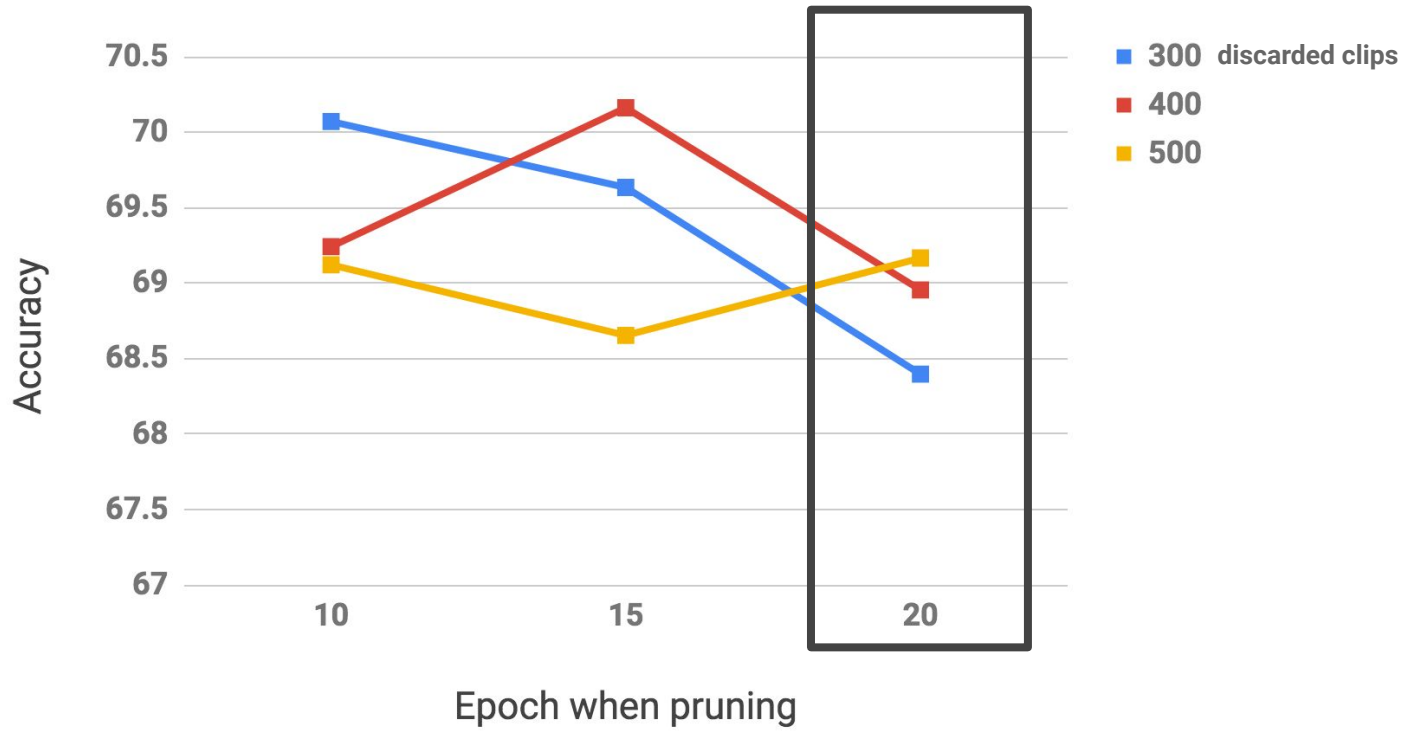
Dataset pruning & noise memorization

- model is more accurate ➜ allows larger pruning (to a certain extent)



Dataset pruning & noise memorization

- model start to memorize noise?



Why this vocabulary?

- data availability
- classes “suitable” for the study of label noise
  - → classes described with tags also used for other audio materials
    - Bass guitar, Crash cymbal, Engine, …
  - → field-recordings: several sound sources expected
    - only the most predominant(s) tagged: Rain, Fireworks, Slam, Fire, …
  - → pairs of related classes:
    - Squeak & Slam / Wind & Rain

Acoustic guitar / Bass guitar / Clapping / Coin (dropping) / Crash cymbal / Dishes, pots, and pans / Engine / Fart / Fire / Fireworks / Glass / Hi-hat / Piano / Rain / Slam / Squeak / Tearing / Walk, footsteps / Wind / Writing